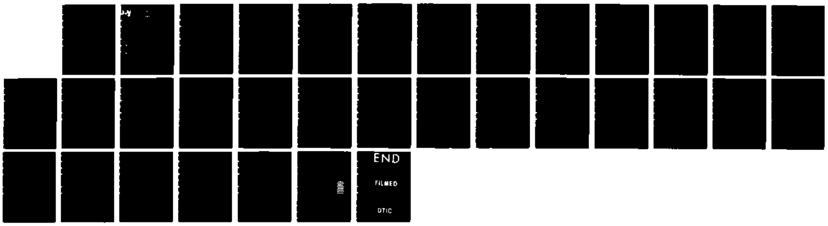
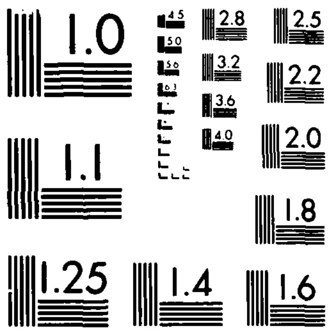
AD-A161 035 A USER'S MANUAL FOR THE FIT PROGRAM: A MODEL FOR SMOOTHING RAW DATA(U) ARMY BALLISTIC RESEARCH LAB ABERDEEN PROVING GROUND MD D M PETRO OCT 85 1/1

UNCLASSIFIED BRL-MR-3472 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD

12

MEMORANDUM REPORT BRL-MR-3472

# A USER'S MANUAL FOR THE FIT PROGRAM: A MODEL FOR SMOOTHING RAW DATA

Denice M. Petro

October 1985

US ARMY BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| | | |
|---|---|---|
| **REPORT DOCUMENTATION PAGE** | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
| 1. REPORT NUMBER MEMORANDUM REPORT BRL-MR-3472 | 2. GOVT ACCESSION NO. AD-A161035 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) A USER'S MANUAL FOR THE FIT PROGRAM: A MODEL FOR SMOOTHING RAW DATA | | 5. TYPE OF REPORT & PERIOD COVERED |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) DENICE M. PETRO | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Ballistic Research Laboratory ATTN: SLCBR-SE ABERDEEN PROVING GROUND, MD 21005-5066 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1L162618AH80 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS US Army Ballistic Research Laboratory ATTN: SLCBR-DD-T ABERDEEN PROVING GROUND, MD 21005-5066 | | 12. REPORT DATE October 1985 |
| | | 13. NUMBER OF PAGES 32 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Analysis
Smoothing
Time Series Analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

FIT is a Fortran 77 program which fits a smooth curve to raw data. The program finds the trends inherent in the data (the peaks, valleys, slopes, etc.) and produces a set of modified data points, which can be plotted using any available graphics package. It is easily used without much knowledge of the underlying techniques and can be employed repeatedly to the subsequent smoothed data sets to achieve any smoothness desired. However, some care must be taken, since the input data must be equally spaced in the independent variable with

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

only a few missing data points. This report covers input preparation, output expectation, technique explanation and provides a program listing in the appendix.

## TABLE OF CONTENTS

A-1

## LIST OF ILLUSTRATIONS

Page

## LIST OF TABLES

## INTRODUCTION

FIT is a Fortran 77 program which fits a smooth curve to raw data. The program finds the trends inherent in the data (the peaks, valleys, slopes, etc.) and produces a set of modified data points, which can be plotted using any available graphics package. Figure 1 illustrates the improvement this program can make. The scatter plot represents the raw data before using the program; while, the curve represents the smoothed data afterwards. However, care must be taken in using this program, since the input data must meet certain criteria explained in a later section.

This report is intended to teach the use of the program. It is structured to guide the reader through input preparation (with sample inputs provided), output expectation (sample outputs provided, as well), and technique explanation. No user manual would be complete without a program listing located in the appendix.

## INPUT

Basically, the data file is composed of an options line, maybe another line with a single number, followed by N (up to 750) data points, one pair per line. The program expects the points to be sequential, i.e., the independent variable of each is incremented by a constant between pairs. Even though a few data points are missing, the program can still be used; however, these points need to be filled in (which the program can do for you when you select a certain option). Let us begin by explaining the various options.

The program can be run in various modes by selecting options from the following list: "e," "f," "n," "s," and/or "x." The "e" option echoes the input data; while the "f" option fills in missing data. A smoothing of the errors is not to be done with the "n" option; while smoothing of the data and the errors is to be done with the "s" option. The "x" option causes extra printing of the partially smoothed data after each step of the process. All five options can be chosen in many combinations; however, the "n" option is useless unless the "s" option has been chosen. The options may appear anywhere in the first 40 columns of the first line of the data file in lower case, in any order, separated by blanks or commas, as desired. Single quotes are not necessary, e.g., esx is a legitimate input line.

The data points should be of the form (independent variable, dependent variable). Both the independent and dependent variables must contain decimal points and must be separated by a comma, e.g., (123.,456.789). If no data points are missing, it is not necessary to list each independent variable; instead, a zero or any other number can be substituted (0.,456.789).

Table 1 lists an example of an input file where no data points are missing; in particular, this is the file used to generate Figure 1. The first line lists the options. For this case, we want the program to echo the input and smooth both the data and the errors. Lines 2 through 30 list the 29 sets of data points in proper format. It is not necessary to count the data points; the program automatically counts them and stops

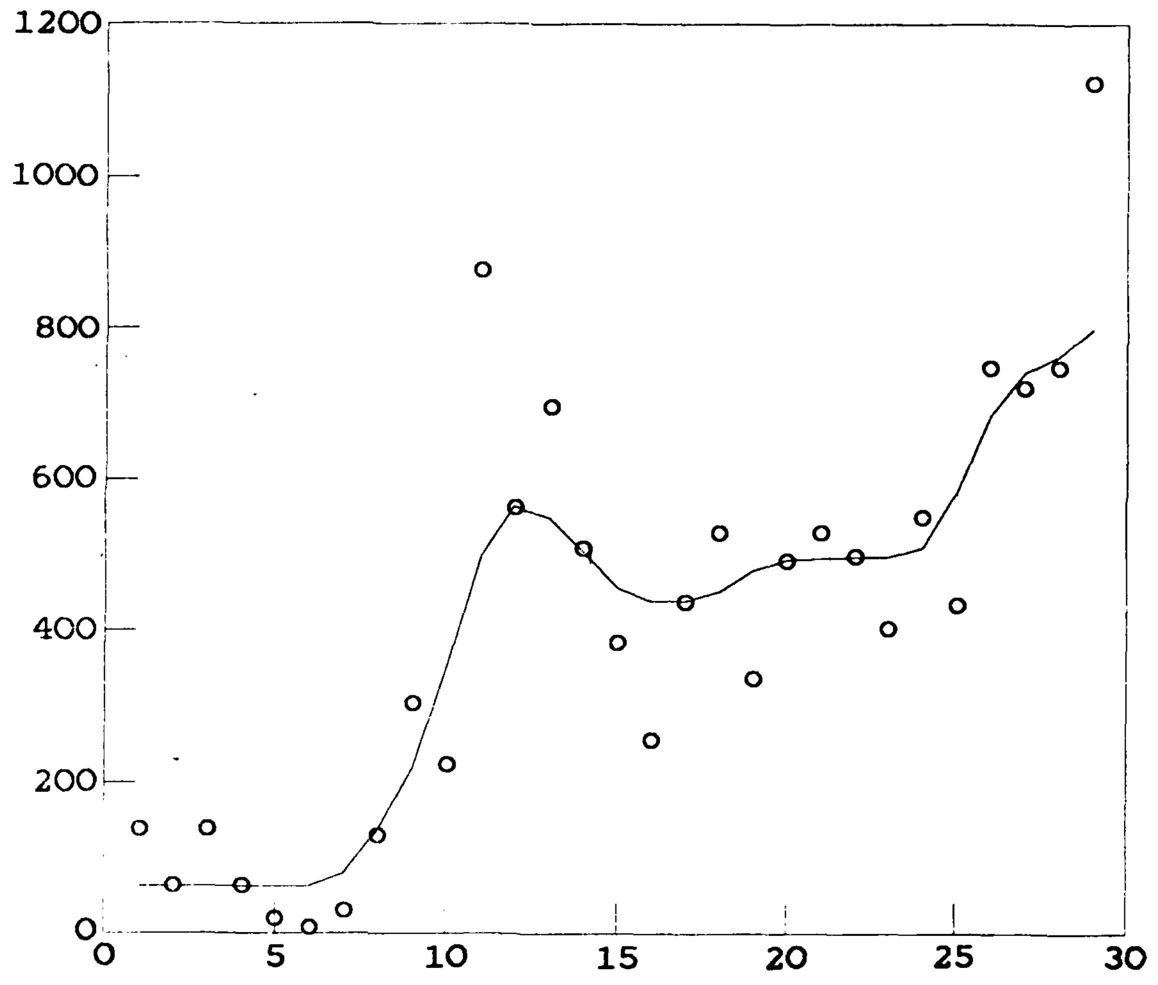


Figure 1. Data Before and After Using the FIT Program

reading data when it reaches end-of-file.

Table 2 gives another example. In this instance, some of the data are missing. Line one of the file asks for the missing points to be filled in as well as for the input to be echoed, for the data to be smoothed, and for the errors not to be smoothed. The second line, one number which must contain a decimal point, specifies the maximum number of data points to be filled in between raw data values. (Please note, this number is not the total number of missing points.) This line is used only when the "f" option is chosen -- don't forget it. Lines 3 through 27 list the sets of data points.

Table 1. Sample Data File I (No Missing Values)

| Line Number | Data in File | Explanation |
|---|---|---|
| 01 | es | Options |
| 02 | 1.,138., | Data Points |
| 03 | 2.,62., | " |
| 04 | 3.,138., | " |
| 05 | 4.,63., | " |
| 06 | 5.,20., | " |
| 07 | 6.,4., | " |
| 08 | 7.,29., | " |
| 09 | 8.,130., | " |
| 10 | 9.,304., | " |
| 11 | 10.,223., | " |
| 12 | 11.,876., | " |
| 13 | 12.,563., | " |
| 14 | 13.,696., | " |
| 15 | 14.,508., | " |
| 16 | 15.,384., | " |
| 17 | 16.,255., | " |
| 18 | 17.,439., | " |
| 19 | 18.,529., | " |
| 20 | 19.,336., | " |
| 21 | 20.,492., | " |
| 22 | 21.,530., | " |
| 23 | 22.,498., | " |
| 24 | 23.,403., | " |
| 25 | 24.,550., | " |
| 26 | 25.,434., | " |
| 27 | 26.,747., | " |
| 28 | 27.,721., | " |
| 29 | 28.,747., | " |
| 30 | 29.,1122., | " |

Table 2. Sample Data File II (Missing Values)

| Line Number | Data in File | Explanation |
|---|---|---|
| 01 | f e s n | Options |
| 02 | 5. | Between raw, max # filled-in |
| 03 | 1.,138., | Data Points |
| 04 | 2.,62., | " |
| 05 | 3.,138., | " |
| 06 | 5.,20., | " |
| 07 | 6.,4., | " |
| 08 | 7.,29., | " |
| 09 | 8.,130., | " |
| 10 | 9.,304., | " |
| 11 | 10.,223., | " |
| 12 | 11.,876., | " |
| 13 | 12.,563., | " |
| 14 | 13.,696., | " |
| 15 | 16.,255., | " |
| 16 | 17.,439., | " |
| 17 | 18.,529., | " |
| 18 | 19.,336., | |
| 19 | 20.,492., | " |
| 20 | 22.,498., | " |
| 21 | 23.,403., | " |
| 22 | 24.,550., | " |
| 23 | 25.,434., | " |
| 24 | 26.,747., | " |
| 25 | 27.,721., | " |
| 26 | 28.,747., | " |
| 27 | 29.,1122., | " |

## OUTPUT

The output files vary depending on the options chosen as well as on the data. In general, the first few lines describe the options; while, ensuing lines list the subroutines as entered, an explanation of the procedure, and the data at various stages. Table 3 gives the output for Sample Data File I from Table 1. The smoothed data you would use for further study is listed under the column "new x(i)" near the end of the listing. The column labeled "orig x(i)" list the original input data and "err (i)" represents the errors between the original and the new data. The "Sigma of err(i)" is the standard deviation of the errors.

Table 4 gives the output for Sample Data File II. Now, when the program enters the input subroutine, it first reads a value, "m0," which represents the maximum number of filled-in points between real data points, before it reads the data points.

Table 3. Output File for Sample Data File I

```
 Echoing input
 Smoothing data
In input subroutine
     i      t(i)         x(i)
     0  1.0000e+00  1.3800e+02
     1  2.0000e+00  6.2000e+01
     2  3.0000e+00  1.3800e+02
     3  4.0000e+00  6.3000e+01
     4  5.0000e+00  2.0000e+01
     5  6.0000e+00  4.0000e+00
     6  7.0000e+00  2.9000e+01
     7  8.0000e+00  1.3000e+02
     8  9.0000e+00  3.0400e+02
     9  1.0000e+01  2.2300e+02
    10  1.1000e+01  8.7600e+02
    11  1.2000e+01  5.6300e+02
    12  1.3000e+01  6.9600e+02
    13  1.4000e+01  5.0800e+02
    14  1.5000e+01  3.8400e+02
    15  1.6000e+01  2.5500e+02
    16  1.7000e+01  4.3900e+02
    17  1.8000e+01  5.2900e+02
    18  1.9000e+01  3.3600e+02
    19  2.0000e+01  4.9200e+02
    20  2.1000e+01  5.3000e+02
    21  2.2000e+01  4.9800e+02
    22  2.3000e+01  4.0300e+02
    23  2.4000e+01  5.5000e+02
    24  2.5000e+01  4.3400e+02
    25  2.6000e+01  7.4700e+02
    26  2.7000e+01  7.2100e+02
    27  2.8000e+01  7.4700e+02
    28  2.9000e+01  1.1220e+03
In smoothing routine
 Perform 3RSSH smoothing on the x(i) values,
 i.e., smooth by repeated medians of 3 (3R)
 up to 3 times, split two-point peaks and valleys
 twice (doing 3R after each split), then hann.
 Subtracting the smoothed data from the original data
 yields some errors -- called the rough
- In median routine
- In median routine
- In median routine
- In splitting routine
- In median routine
- In median routine
```

Table 3. Output File for Sample Data File I (cont'd)

```
- In splitting routine
- In median routine
- In median routine
- In hanning routine
    i   original      smooth        rough
    0  1.3800e+02  6.3000e+01  7.5000e+01
    1  6.2000e+01  6.3000e+01 -1.0000e+00
    2  1.3800e+02  6.3000e+01  7.5000e+01
    3  6.3000e+01  6.3000e+01  0.    e+00
    4  2.0000e+01  6.3000e+01 -4.3000e+01
    5  4.0000e+00  6.3000e+01 -5.9000e+01
    6  2.9000e+01  7.9750e+01 -5.0750e+01
    7  1.3000e+02  1.3650e+02 -6.5000e+00
    8  3.0400e+02  2.2000e+02  8.4000e+01
    9  2.2300e+02  3.4850e+02 -1.2550e+02
   10  8.7600e+02  4.9825e+02  3.7775e+02
   11  5.6300e+02  5.6300e+02  0.    e+00
   12  6.9600e+02  5.4925e+02  1.4675e+02
   13  5.0800e+02  5.0450e+02  3.5000e+00
   14  3.8400e+02  4.5625e+02 -7.2250e+01
   15  2.5500e+02  4.3900e+02 -1.8400e+02
   16  4.3900e+02  4.3900e+02  0.    e+00
   17  5.2900e+02  4.5225e+02  7.6750e+01
   18  3.3600e+02  4.7875e+02 -1.4275e+02
   19  4.9200e+02  4.9350e+02 -1.5000e+00
   20  5.3000e+02  4.9650e+02  3.3500e+01
   21  4.9800e+02  4.9800e+02  0.    e+00
   22  4.0300e+02  4.9800e+02 -9.5000e+01
   23  5.5000e+02  5.1100e+02  3.9000e+01
   24  4.3400e+02  5.7975e+02 -1.4575e+02
   25  7.4700e+02  6.8475e+02  6.2250e+01
   26  7.2100e+02  7.4050e+02 -1.9500e+01
   27  7.4700e+02  7.6000e+02 -1.3000e+01
   28  1.1220e+03  7.9900e+02  3.2300e+02
 Now, smooth the rough with 3RSSH
- In median routine
- In median routine
- In median routine
- In splitting routine
- In median routine
- In median routine
- In splitting routine
- In median routine
- In median routine
- In hanning routine
```

Table 3. Output File for Sample Data File I (cont'd)

```
Re-rough the initial smooth by adding the smooth of rough
i.e., new x(i) = smooth of smooth + smooth of rough
    i   orig x(i)   new x(i)     err(i)
    0  1.3800e+02  6.3000e+01  7.5000e+01
    1  6.3000e+01  6.3000e+01 -1.0000e+00
    2  1.3800e+02  6.3000e+01  7.5000e+01
    3  6.3000e+01  6.3000e+01  0.    e+00
    4  2.0000e+01  6.1375e+01 -4.1375e+01
    5  4.0000e+00  5.8125e+01 -5.4125e+01
    6  2.9000e+01  7.3250e+01 -4.4250e+01
    7  1.3000e+02  1.3000e+02  0.    e+00
    8  3.0400e+02  2.1513e+02  8.8875e+01
    9  2.2300e+02  3.4775e+02 -1.2475e+02
   10  8.7600e+02  5.0088e+02  3.7513e+02
   11  5.6300e+02  5.6650e+02 -3.5000e+00
   12  6.9600e+02  5.5275e+02  1.4325e+02
   13  5.0800e+02  5.0800e+02  0.    e+00
   14  3.8400e+02  4.5888e+02 -7.4875e+01
   15  2.5500e+02  4.3988e+02 -1.8488e+02
   16  4.3900e+02  4.3900e+02  0.    e+00
   17  5.2900e+02  4.5225e+02  7.6750e+01
   18  3.3600e+02  4.7875e+02 -1.4275e+02
   19  4.9200e+02  4.9350e+02 -1.5000e+00
   20  5.3000e+02  4.9650e+02  3.3500e+01
   21  4.9800e+02  4.9800e+02  0.    e+00
   22  4.0300e+02  4.9800e+02 -9.5000e+01
   23  5.5000e+02  5.0775e+02  4.2250e+01
   24  4.3400e+02  5.7000e+02 -1.3600e+02
   25  7.4700e+02  6.7175e+02  7.5250e+01
   26  7.2100e+02  7.2750e+02 -6.5000e+00
   27  7.4700e+02  7.4700e+02  0.    e+00
   28  1.1220e+03  7.8600e+02  3.3600e+02
Sigma of err(i) =    122.131
```

Table 4. Output File for Sample Data File II

```
 Echoing input
 Equalizing data by filling in
 Smoothing data
 Not re-roughing the initial smoothed data
In input subroutine
 max number of interpolated points  between real data points, m0 =   5.00000
     i     t(i)        x(i)
     0  1.0000e+00  1.3800e+02
     1  2.0000e+00  6.2000e+01
     2  3.0000e+00  1.3800e+02
     3  5.0000e+00  2.0000e+01
     4  6.0000e+00  4.0000e+00
     5  7.0000e+00  2.9000e+01
     6  8.0000e+00  1.3000e+02
     7  9.0000e+00  3.0400e+02
     8  1.0000e+01  2.2300e+02
     9  1.1000e+01  8.7600e+02
    10  1.2000e+01  5.6300e+02
    11  1.3000e+01  6.9600e+02
    12  1.6000e+01  2.5500e+02
    13  1.7000e+01  4.3900e+02
    14  1.8000e+01  5.2900e+02
    15  1.9000e+01  3.3600e+02
    16  2.0000e+01  4.9200e+02
    17  2.2000e+01  4.9800e+02
    18  2.3000e+01  4.0300e+02
    19  2.4000e+01  5.5000e+02
    20  2.5000e+01  4.3400e+02
    21  2.6000e+01  7.4700e+02
    22  2.7000e+01  7.2100e+02
    23  2.8000e+01  7.4700e+02
    24  2.9000e+01  1.1220e+03
In fill subroutine
     i  filled t(i) filled x(i)
     0  1.0000e+00  1.3800e+02
     1  2.0000e+00  6.2000e+01
     2  3.0000e+00  1.3800e+02
     3  4.0000e+00  7.9000e+01
     4  5.0000e+00  2.0000e+01
     5  6.0000e+00  4.0000e+00
     6  7.0000e+00  2.9000e+01
     7  8.0000e+00  1.3000e+02
     8  9.0000e+00  3.0400e+02
     9  1.0000e+01  2.2300e+02
    10  1.1000e+01  8.7600e+02
    11  1.2000e+01  5.6300e+02
```

Table 4. Output File for Sample Data File II (cont'd)

```
     12  1.3000e+01  6.9600e+02
     13  1.4000e+01  5.4900e+02
     14  1.5000e+01  4.0200e+02
     15  1.6000e+01  2.5500e+02
     16  1.7000e+01  4.3900e+02
     17  1.8000e+01  5.2900e+02
     18  1.9000e+01  3.3600e+02
     19  2.0000e+01  4.9200e+02
     20  2.1000e+01  4.9500e+02
     21  2.2000e+01  4.9800e+02
     22  2.3000e+01  4.0300e+02
     23  2.4000e+01  5.5000e+02
     24  2.5000e+01  4.3400e+02
     25  2.6000e+01  7.4700e+02
     26  2.7000e+01  7.2100e+02
     27  2.8000e+01  7.4700e+02
     28  2.9000e+01  1.1220e+03
In smoothing routine
 Perform 3RSSH smoothing on the x(i) values,
 i.e., smooth by repeated medians of 3 (3R)
 up to 3 times, split two-point peaks and valleys
 twice (doing 3R after each split), then hann.
 Subtracting the smoothed data from the original data
 yields some errors -- called the rough
- In median routine
- In median routine
- In median routine
- In splitting routine
- In median routine
- In median routine
- In splitting routine
- In median routine
- In median routine
- In hanning routine
      i   original      smooth        rough
      0  1.3800e+02  7.9000e+01  5.9000e+01
      1  6.2000e+01  7.9000e+01 -1.7000e+01
      2  1.3800e+02  7.9000e+01  5.9000e+01
      3  7.9000e+01  7.9000e+01  0.    e+00
      4  2.0000e+01  7.9000e+01 -5.9000e+01
      5  4.0000e+00  7.9000e+01 -7.5000e+01
      6  2.9000e+01  9.1750e+01 -6.2750e+01
      7  1.3000e+02  1.4050e+02 -1.0500e+01
      8  3.0400e+02  2.2000e+02  8.4000e+01
      9  2.2300e+02  3.4850e+02 -1.2550e+02
```

Table 4.  Output File for Sample Data File II (cont'd)

```
   10  8.7600e+02  4.9825e+02  3.7775e+02
   11  5.6300e+02  5.6300e+02  0.     e+00
   12  6.9600e+02  5.5950e+02  1.3650e+02
   13  5.4900e+02  5.4550e+02  3.5000e+00
   14  4.0200e+02  5.0750e+02 -1.0550e+02
   15  2.5500e+02  4.5950e+02 -2.0450e+02
   16  4.3900e+02  4.3900e+02  0.     e+00
   17  5.2900e+02  4.5225e+02  7.6750e+01
   18  3.3600e+02  4.7875e+02 -1.4275e+02
   19  4.9200e+02  4.9275e+02 -7.5000e-01
   20  4.9500e+02  4.9425e+02  7.5000e-01
   21  4.9800e+02  4.9500e+02  3.0000e+00
   22  4.0300e+02  4.9575e+02 -9.2750e+01
   23  5.5000e+02  5.1025e+02  3.9750e+01
   24  4.3400e+02  5.7975e+02 -1.4575e+02
   25  7.4700e+02  6.8475e+02  6.2250e+01
   26  7.2100e+02  7.4050e+02 -1.9500e+01
   27  7.4700e+02  7.6000e+02 -1.3000e+01
   28  1.1220e+03  7.9900e+02  3.2300e+02
Sigma of err(i) =    123.050
```

Then, the program continues into the fill subroutine where "m0" is tested. If this value had been too small, the program would have printed an error message and stopped (See Table 5). Given that this has not happened, a new set of data points are printed. The original data have not been altered; just additional points have been added. The program continues through the rest of the subroutines as before, but, since we asked for no smoothing of the errors, it stops earlier. Now, the smoothed data to be used for further study is listed under the column "smooth" near the end of the listing; whereas, the column "rough" represents the errors between the smooth and the original data. The standard deviation of the rough is given by "Sigma of err(i)."

Table 5. Sample Output File When Inadequate Fill-In Parameter Given

```
  Echoing input
  Equalizing data by filling in
  Smoothing data
  Not re-roughing the initial smoothed data
 In input subroutine
  max number of interpolated points  between real data points, m0 =   1.00000
     i      t(i)         x(i)
     0  1.0000e+00  1.3800e+02
     1  2.0000e+00  6.2000e+01
     2  3.0000e+00  1.3800e+02
     3  5.0000e+00  2.0000e+01
     4  6.0000e+00  4.0000e+00
     5  7.0000e+00  2.9000e+01
     6  8.0000e+00  1.3000e+02
     7  9.0000e+00  3.0400e+02
     8  1.0000e+01  2.2300e+02
     9  1.1000e+01  8.7600e+02
    10  1.2000e+01  5.6300e+02
    11  1.3000e+01  6.9600e+02
    12  1.6000e+01  2.5500e+02
    13  1.7000e+01  4.3900e+02
    14  1.8000e+01  5.2900e+02
    15  1.9000e+01  3.3600e+02
    16  2.0000e+01  4.9200e+02
    17  2.2000e+01  4.9800e+02
    18  2.3000e+01  4.0300e+02
    19  2.4000e+01  5.5000e+02
    20  2.5000e+01  4.3400e+02
    21  2.6000e+01  7.4700e+02
    22  2.7000e+01  7.2100e+02
    23  2.8000e+01  7.4700e+02
    24  2.9000e+01  1.1220e+03
 In fill subroutine
  Equalizing step size is impossible;
  3.00000   is greater than m0 =   1.00000
```

## METHODOLOGY

The program described above uses the 3RSSH method of Tukey for smoothing the data.[1] First, we will decipher the acronym and, then, describe the techniques. 3R stands for Repeated medians of 3. SS is Splitting of two-point peaks and valleys -- done twice; while H is Hanning.

With 3R, the program takes the first three dependent values (0, 1, and 2), finds the median, and stores that in a new array as point 1. Next, it takes points 1 through 3, finds the median, and stores that as new point 2. This process continues until it takes points N-2, N-1, and N and stores their median in point N-1. To complete the new array, the program just copies points 0 and N onto it. If this new array is equal to the original array, 3R is stopped; otherwise, this process continues until no changes take place or a total of three times, whichever comes first. However, the second iteration starts with point 1 and continues through N-1 (copying on points 0, 1, N-1, and N); while the third iteration starts at point 2 and goes to N-2 (copying points 0, 1, 2, N-2, N-1, and N). Whenever 3R is completed, these copied-on, end values need smoothing which is done by finding the median of the end value, the last smoothed value, and the difference between thrice the last smoothed value and twice the next-to-last smoothed value. For example, if 3R has been performed three times, the points 0, 1, 2, N-2, N-1, and N all need smoothed. For point 0, the formula would be:

new point 0 = median(point 0, point 3, 3*(point 3) - 2*(point 4)).

For points 1 and 2, the same formula applies with just the substitution of these points for point 0. To find points N-2, N-1 and N, just substitute each of these, in turn, for point 0 and change point 3 to point N-3 and point 4 to point N-4.

To perform the splitting portion of 3RSSH, the program examines the 3R data to see if there are any peaks or valleys consisting of just two, equal data points. If so, they are split apart and end-point smoothing (the same formula as previous paragraph) is performed on each point as if it were the last. In this case, however, the last and next-to-last smoothed data points are now adjacent to the end points, i. e., if the peak values are i and i+1, then we would use points i, i-1, and i-2 to determine the new point i, and i+1, i+2, i+3 to determine the new point i+1. When all such pairs have been examined, another 3R is done on the data. This entire process is repeated with another split and 3R. However, if there were no splits, the program immediately goes to the hanning subroutine, since no changes have occurred in the data.

In the hanning technique, the program, first, runs through the data set and finds the mid-range between every other data point, which becomes a tentative new value for the intermediate data point. Then, it finds the mean of the original, intermediate value and the tentative value. This mean becomes the newest value for each data point in the set. For example, let the original data set be the values 4, 5, 6, 8, 11. The mid-ranges would be (4+6)/2, (5+8)/2, (6+11)/2. The means, the averages of the mid-range and

[1] *John W. Tukey, Exploratory Data Analysis, Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1977, pp 204-286 and 523-542.*

the original number, would be (5+5)/2, (6.5+6)/2, (8.5+8)/2. Therefore, the final set of values would be 4, 5, 6.25, 8.25, 11.

Hanning finishes the 3RSSH process and its resultant data set is called the smooth. This smooth is then subtracted from the original data to obtain the errors, also known as the rough. To obtain a better smooth, it is a good idea to perform 3RSSH on this rough. The smooth we obtain from the rough is then added to the first smooth yielding a final smooth and a final rough.

## SUMMARY

The FIT program enables you to smooth sequential data to observe the inherent trends. It is easily used without much knowledge of the underlying techniques and can be employed repeatedly to the subsequent smoothed data sets to achieve whatever smoothness is desired.

## REFERENCES

1. John W. Tukey, Exploratory Data Analysis, Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1977, pp 204-236 and 523-542.

## Appendix: Program Listing

```
c       Purpose: Fit data
        logical ECHO, FILLIN, SMOOTH, XTRA, NRERUF
        logical isarg
        character*1 arg
        dimension arg(40)
        common /misc/ ECHO, FILLIN, XTRA, NRERUF
1       format(40a1)
c
        read(5,1) arg
        ECHO   = isarg('e',arg)
        FILLIN = isarg('f',arg)
        SMOOTH = isarg('s',arg)
        XTRA   = isarg('x',arg)
        NRERUF = isarg('n',arg)
        if (ECHO  ) print *, ' Echoing input'
        if (FILLIN) print *, ' Equalizing data by filling in'
        if (SMOOTH) print *, ' Smoothing data'
        if (XTRA  ) print *, ' Extra printing of intermediate results'
        if (NRERUF) print *, ' Not re-roughing the initial smoothed data'
        call input(em0)
c       call equalizing routine to fill in missing data
                if (FILLIN) call fill(em0)
        if (SMOOTH) call smoth
        END
c
        LOGICAL FUNCTION IS ARG (ACHAR, ARG)
c       -----------------------------------
c       Return true iff achar is in arg
        character*1 achar
        character*1 arg
        dimension arg(40)
c
        is arg = .FALSE.
c       Loop through each element of arg until ACHAR is found.
            DO 30 narg = 1,40
                IF (ACHAR .eq. arg(narg)) THEN
                    is arg = .TRUE.
                                                                GOTO 99
                ENDIF
30          CONTINUE
99      CONTINUE
        END
c
        SUBROUTINE INPUT ( em0 )
c       ----------------------
        logical sorted, ECHO, FILLIN, XTRA, NRERUF
        common/data/tt(0:749),x(0:749),n
        common /misc/ ECHO, FILLIN, XTRA, NRERUF
1       format (i8,2(1pe12.4))
        print *,'In input subroutine'
2       format(2f12.4)
```

```
c       if FILLIN is true (FILL is used), then
c       read em0, where em0 is the maximum number off interpolated
c       points between real data points
        IF (FILLIN) THEN
            read 2, em0
            if (ECHO) print *, ' max number of interpolated points'
     1          ,'between real data points, m0 =', em0
        ENDIF
        if (ECHO) print *, '     i      t(i)          x(i)'
        DO 10 i = 0,749
                read(5,2,END=500) tt(i), x(i)
                if (ECHO) print 1, i,tt(i),x(i)
10      CONTINUE
500     n = i - 1
        DO 20 i = 1,n-1
                sorted = .true.
                DO 30 nn = 0,n-1
                        if (tt(nn) .le. tt(nn+1)) go to 30
                        temp = tt(nn)
                        tt(nn)  = tt(nn+1)
                        tt(nn+1) = temp
                        temp = x(nn)
                        x(nn)  = x(nn+1)
                        x(nn+1) = temp
                        sorted = .false.
30              CONTINUE
                if (sorted) go to 40
20      CONTINUE
40      CONTINUE
        delta = (tt(n) - tt(0)) / n
        END
c
        SUBROUTINE FILL ( em0 )
c       -----------------------
        logical sorted, ECHO, FILLIN, XTRA, NRERUF
        integer p
        dimension deltan(750),z(0:749),t(0:750),tp(0:749)
        common/data/tt(0:749),x(0:749),n
        common /misc/ ECHO, FILLIN, XTRA, NRERUF
        print *,'In fill subroutine'
c  ***** store independent variable tt into t since t needs to be
c  ***** one longer
        DO 5 nn = 0,n
                t(nn) = tt(nn)
5       CONTINUE
c  ***** find the differences between each indep. variable
        DO 10 nn = 1,n
                deltan(nn) = t(nn) - t(nn-1)
10      CONTINUE
c  ***** now sort these differences
        DO 20 i = 1,nn-1
                sorted = .true.
                DO 30 nn = 1,n-1
                        if (deltan(nn) .le. deltan(nn+1)) go to 30
                        temp = deltan(nn)
                        deltan(nn)  = deltan(nn+1)
                        deltan(nn+1) = temp
                        sorted = .false.
30              CONTINUE
                if (sorted) go to 40
20        CONTINUE
c  ***** now find median and maximum of these "delta's"
40        IF (mod(n,2) .ne. 0) THEN
          delmed = deltan(n/2 + 1)
          ELSE
          delmed = (deltan(n/2) + deltan(n/2 + 1)) / 2.
          ENDIF
          delmax = deltan(n)
          ncap1 = (t(n) - t(0))/delmed
          if (ncap1 .ne. (t(n) - t(0))/delmed) ncap1 = ncap1 + 1
          delta = (t(n) - t(0)) / ncap1
          t(n+1) = 1.e20
          if (delmax/delta .gt. em0) go to 100
c  ***** equalize step size and fill in data
          DO 50 p = 0, ncap1
                tp(p) = t(0) + p*delta
                DO 60 nn = 0,n
                if((t(nn) .le. tp(p)) .and. (tp(p) .lt. t(nn+1))) go to 70
60              CONTINUE
70              z(p) = ((tp(p) -t(nn))*x(nn+1) + (t(nn+1) -tp(p))*x(nn)) /
     1                   (t(nn+1) -t(nn))
50        CONTINUE
          n = ncap1
          print *, '     i  filled t(i) filled x(i)'
          DO 80 nn = 0,n
                x(nn) = z(nn)
                print 1, nn,tp(nn),x(nn)
80        CONTINUE
          RETURN
100       dlmdl = delmax/delta
          print *, ' Equalizing step size is impossible; '
          print *, dlmdl,
     1    ' is greater than m0 =',em0
          STOP
1         format(i8,2(1pe12.4))
          END
c
          SUBROUTINE SMOTH
c         ----------------
          logical done, ECHO, FILLIN, XTRA, NRERUF
          dimension y(0:749), xhat(0:749), err(0:749), z(0:749)
          common/data/tt(0:749),x(0:749),n
          common /misc/ ECHO, FILLIN, XTRA, NRERUF
1         format (i8,3(1pe12.4))
c
          print *,'In smoothing routine'
          print *, ' Perform 3RSSH smoothing on the x(i) values,'
          print *, ' i.e., smooth by repeated medians of 3 (3R)'
          print *, ' up to 3 times, split two-point peaks and valleys'
          print *, ' twice (doing 3R after each split), then hann.'
          print *, ' Subtracting the smoothed data from the original data'
          print *, ' yields some errors -- called the rough'
c         Store a copy of x into y
          call copy(n,x,y)
c         Smooth by medians 3 times or until no more changes (3R)
          done = .false.
          DO 10 i = 1,3
              IF (done)   GOTO 15
              call median(n,y,done,i)
10        CONTINUE
15        CONTINUE
c         Split twice or until no more splits (SS)
          done = .false.
          DO 20 nsplit = 1,2
              call split(done,n,y)
          IF (XTRA) THEN
              print *, '     i   new value'
              DO 22 ijk=0,n
                  print 1, ijk, y(ijk)
22            CONTINUE
          ENDIF
              IF (done)   GOTO 25
c         If there's no splits then exit loop
c         Otherwise 3R the new split values
              DO 23 i=1,3
                  IF (done) GOTO 24
                  call median(n,y,done,i)
23            CONTINUE
24            CONTINUE
              done = .false.
20        CONTINUE
25        CONTINUE
          IF (XTRA) THEN
              print *, '     i   new value'
              DO 27 ijk=0,n
                  print 1, ijk, y(ijk)
27            CONTINUE
          ENDIF
c         Hann the 3RSS data (H)
              call haning(n,y,z)
c         Store this 3RSSH smoothed data into xhat
              call copy(n,z,xhat)
c         Find the rough of the smoothed data
c         And, print the original and 3RSSH smooth & rough
              print *, '     i   original     smooth        rough'
          sum sq = 0.
              DO 30 i=0,n
                  y(i) = x(i)-xhat(i)
                  if (NRERUF) sum sq = sum sq + y(i)**2
                  print 1, i, x(i), xhat(i), y(i)
30            CONTINUE
          IF ( .not. NRERUF) THEN
```

```
c       3RSS and Hann the rough of initial 3RSSH
c       Smooth by medians 3 times or until no more changes
            print *, ' Now, smooth the rough with 3RSSH'
            done = .false.
            DO 32 i = 1,3
                IF (done)       GOTO 34
                call median(n,y,done,i)
32          CONTINUE
34          CONTINUE
c       Split twice or until no more splits
            done = .false.
            DO 35 nsplit = 1,2
                call split(done,n,y)
        IF (XTRA) THEN
                print *, '      i    new value'
                DO 36 ijk=0,n
                        print 1, ijk, y(ijk)
36              CONTINUE
        ENDIF
                IF (done)       GOTO 39
c       If there's no splits then exit loop
c       Otherwise 3R the new split values
                DO 37 i=1,3
                IF (done) GOTO 38
                call median(n,y,done,i)
37              CONTINUE
38              CONTINUE
                done = .false.
35          CONTINUE
39          CONTINUE
        IF (XTRA) THEN
            print *, '      i    new value'
            DO 41 ijk=0,n
                print 1, ijk, y(ijk)
41          CONTINUE
        ENDIF
c       Hann the 3RSS of the rough
            call haning(n,y,z)
        IF (XTRA) THEN
        print *, '      i smooth of rough'
            DO 43 ijk=0,n
                print 1, ijk, z(ijk)
43          CONTINUE
        ENDIF
            sum sq = 0.
c       Re-rough by adding the smooth of the rough to the initial smooth
c       to find the final smooth
c       and find the error between orig. data and final smooth
        print *, ' Re-rough the initial smooth by adding the smooth of rough'
        print *,' i.e., new x(i) = smooth of smooth + smooth of rough'
        print *, '      i    orig x(i)    new x(i)     err(i)'
            DO 40 i=0,n
                xhat(i) = xhat(i) + z(i)
                err(i) = x(i) - xhat(i)
```

```
                 sum sq = sum sq+err(i)**2
                 print 1, i, x(i), xhat(i), err(i)
                 x(i) = xhat(i)
40            CONTINUE
          ENDIF
              sigma s = sqrt(sum sq/n)
              print *, 'Sigma of err(i) = ', sigma s
          END
c
          SUBROUTINE MEDIAN(n,y,done,k)
c         ----------------------------
c         Smooth by taking medians of 3
          real medin
          logical done,mdone,ECHO,FILLIN,XTRA,NRERUF
          common /misc/ ECHO, FILLIN, XTRA, NRERUF
          dimension y(0:749),z(0:749)
1         format (i8,2(1pe12.4))
c
          print *,'- In median routine'
          mdone = .true.
c         Copy on the end input values
          DO 5 i = 0,k-1
              z(i) = y(i)
5         CONTINUE
          DO 10 i = n-k+1,n
              z(i) = y(i)
10        CONTINUE
c         Start the median-taking at the k-th element
          DO 20 i=k,n-k
              z(i) = medin(y(i-1),y(i),y(i+1))
              if( z(i) .ne. y(i) ) mdone = .false.
20        CONTINUE
c         Smooth end input values when medians of 3 are finished
c         either when no more changes have taken place or when
c         k reaches 3
          IF ( (mdone) .or. (k .eq. 3) ) THEN
              DO 32 i = 0,k-1
                  z(i) = medin(3*z(k)-2*z(k+1),y(i),z(k))
32            CONTINUE
              DO 37 i = n-k+1,n
                  z(i) = medin(z(n-k),y(i),3*z(n-k)-2*z(n-k-1))
37            CONTINUE
          ENDIF
          if (XTRA) print *, '     i    old value    new value'
          DO 40 i=0,n
              if (XTRA) print 1, i, y(i), z(i)
              y(i) = z(i)
40        CONTINUE
          done = mdone
          END
c
          SUBROUTINE SPLIT(done,n,y)
c         -------------------------
c         perform splitting of data at plateaus
```

```
        logical done,nsplit
        real medin
        dimension y(0:749),z(0:749)
c
        print *,'- In splitting routine'
        nsplit = .true.
c  ** change from do-loop because incrementing of loop index
c  ** while in the loop is not allowed by all compilers
c       DO 10 i=2,n-3
        i = 2
c       if data not equal leave alone
7               if(y(i) .eq. y(i+1)) GO TO 5
2               z(i) = y(i)
                GO TO 10
c       if two adjacent data points equal and slope continuing in
c       same direction; leave alone
5               if ( (y(i) .eq. y(i+2)) .or. (y(i) .eq. y(i-1)) ) GOTO 2
                if(sign(1.,y(i)-y(i-1)).ne.sign(1.,y(i+1)-y(i+2))) GO TO 2
c       else
c       smooth out plateau by giving these two points new values
c       also change nsplit to indicate a splitting
                nsplit = .false.
                z(i) = medin(3*y(i-1)-2*y(i-2),y(i-1),y(i))
                z(i+1) = medin(3*y(i+2)-2*y(i+3),y(i+2),y(i+1))
                i = i+1
10      CONTINUE
c  ** next two lines replace do-loop increment and test
        i = i + 1
        if (i .le. n-3) goto 7
        DO 20 i=2,n-3
                y(i) = z(i)
20      CONTINUE
        if(nsplit) done = .true.
        END
c
        SUBROUTINE HANING (n,y,z)
c       ------------------------
c       Perform hanning's smoothing method
        dimension y(0:749), z(0:749)
c
        print *,'- In hanning routine'
        DO 20 i=1,n-1
            z(i) = 0.25*(y(i-1)+2*y(i)+y(i+1))
20      CONTINUE
        z(0) = y(0)
        z(n) = y(n)
        END
c
        SUBROUTINE COPY (n, x, y)
c       ------------------------
c       Copy vector of length n from x to y
        dimension x(0:749), y(0:749)
        DO 20 i=0,n
            y(i) = x(i)
```

```
20      CONTINUE
        END
c
        REAL FUNCTION MEDIN (a, b, c)
c       -----------------------------
c       Find the median of a, b, c
        IF (sign(1.,a-b) .eq. sign(1.,b-c)) THEN
            medin = b
        ELSE IF (abs(a-b) .gt. abs(b-c)) THEN
            medin = c
        ELSE
            medin = a
        ENDIF
        END
```

DISTRIBUTION LIST

| No. of Copies | Organization |
|---|---|
| 12 | Administrator<br>Defense Technical Info Center<br>ATTN: DTIC-DDA<br>Cameron Station<br>Alexandria, VA 22314 |
| 1 | HQDA<br>DAMA-ART-M<br>Washington, DC 20310 |
| 1 | Commander<br>US Army Materiel Command<br>ATTN: AMCDRA-ST<br>5001 Eisenhower Avenue<br>Alexandria, VA 22333-0001 |
| 1 | Commander<br>Armament R&D Center<br>US Army AMCCOM<br>ATTN: SMCAR-TDC<br>Dover, NJ 07801-5001 |
| 1 | Commander<br>Armament R&D Center<br>US Army AMCCOM<br>ATTN: SMCAR-TSS<br>Dover, NJ 07801-5001 |
| 1 | Commander<br>US Army Armament, Munitions &<br>Chemical Command<br>ATTN: SMCAR-ESP-L<br>Rock Island, IL 61299-6000 |
| 1 | Director<br>Benet Weapons Laboratory<br>Armament R&D Center<br>US Army AMCCOM<br>ATTN: SMCAR-LCB-TL<br>Watervliet, NY 12189 |
| 1 | Commander<br>US Army Aviation Research<br>and Development Command<br>ATTN: AMSAV-E<br>4300 Goodfellow Boulevard<br>St. Louis, MO 63120 |
| 1 | Director<br>US Army Air Mobility Research<br>and Development Laboratory<br>Ames Research Center<br>Moffett Field, CA 94035 |
| 1 | Commander<br>US Army Communications-<br>Electronics Command<br>ATTN: AMSEL-ED<br>Fort Monmouth, NJ 07703 |
| **1** | **Commander<br>ERADCOM Technical Library<br>ATTN: DELSD-L (Reports Section)<br>Fort Monmouth, NJ 07703-5301** |
| **1** | **Commander<br>US Army Tank Automotive Command<br>ATTN: AMSTA-TSL<br>Warren, MI 48090** |
| **1** | **Commander<br>US Army Missile Command<br>Research, Development and<br>Engineering Center<br>ATTN: AMSMI-RD<br>Redstone Arsenal, AL 35898** |
| **1** | **Commander<br>US Army Missile and Space<br>Intelligence Center<br>ATTN: AIAMS-YDL<br>Redstone Arsenal, AL 35898-5500** |
| 1 | Director<br>US Army TRADOC Systems<br>Analysis Activity<br>ATTN: ATAA-SL<br>White Sands Missile Range,<br>NM 88002 |
| 1 | Air Force Armament Laboratory<br>ATTN: AFATL/DLODL<br>Eglin AFB, FL 32542-5000 |
| 1 | Commandant<br>US Army Infantry School<br>ATTN: ATSH-CD-CSO-OR<br>Fort Benning, GA 31905 |

DISTRIBUTION LIST

| No. of Copies | Organization |
|---|---|
| 1 | Commander<br>US Army Development & Employment Agency<br>ATTN: MODE-TED-SAB<br>Fort Lewis, WA 98433 |
| 1 | AFWL/SUL<br>Kirtland AFB, NM 87117 |

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, Mr. H. Cohen
Cdr, USATECOM
ATTN: AMSTE-TO-F
Cdr, CRDC, AMCCOM
ATTN: SMCCR-RSP-A
SMCCR-MU
SMCCR-SPS-IL

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number____________________ Date of Report__________

2. Date Report Received____________________

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.)__________

4. How specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.)__________

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided or efficiencies achieved, etc? If so, please elaborate.__________

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

CURRENT ADDRESS

Name

Organization

Address

City, State, Zip

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

Name

Organization

Address

City, State, Zip

(Remove this sheet along the perforation, fold as indicated, staple or tape closed, and mail.)

— — — — — — — — — FOLD HERE — — — — — — — — — —

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**OFFICIAL BUSINESS**
**PENALTY FOR PRIVATE USE, $300**

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO 12062 WASHINGTON, DC
POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989

— — — — — — — — — FOLD HERE — — — — — — — — — —